Spherical Transformer for LiDAR-based 3D Recognition

Xin Lai\textsuperscript{1}  Yukang Chen\textsuperscript{1}  Fanbin Lu\textsuperscript{1}  Jianhui Liu\textsuperscript{2}  Jiaya Jia\textsuperscript{1,3}

\textsuperscript{1}The Chinese University of Hong Kong  \textsuperscript{2}The University of Hong Kong  \textsuperscript{3}SmartMore

Abstract

LiDAR-based 3D point cloud recognition has benefited various applications. Without specially considering the LiDAR point distribution, most current methods suffer from information disconnection and limited receptive field, especially for the sparse distant points. In this work, we study the varying-sparsity distribution of LiDAR points and present \textit{SphereFormer} to directly aggregate information from dense close points to the sparse distant ones. We design radial window self-attention that partitions the space into multiple non-overlapping narrow and long windows. It overcomes the disconnection issue and enlarges the receptive field smoothly and dramatically, which significantly boosts the performance of sparse distant points. Moreover, to fit the narrow and long windows, we propose exponential splitting to yield fine-grained position encoding and dynamic feature selection to increase model representation ability. Notably, our method ranks 1\textsuperscript{st} on both nuScenes and SemanticKITTI semantic segmentation benchmarks with 81.9\% and 74.8\% mIoU, respectively. Also, we achieve the 3\textsuperscript{rd} place on nuScenes object detection benchmark with 72.8\% NDS and 68.5\% mAP. Code is available at https://github.com/dvlab-research/SphereFormer.git.

1. Introduction

Nowadays, point clouds can be easily collected by LiDAR sensors. They are extensively used in various industrial applications, such as autonomous driving and robotics. In contrast to 2D images where pixels are arranged densely and regularly, LiDAR point clouds possess the varying-sparsity property — points near the LiDAR are quite dense, while points far away from the sensor are much sparser, as shown in Fig. 2 (a).

However, most existing work \cite{12, 13, 24, 25, 55, 70–72} does not specially consider the the varying-sparsity point distribution of outdoor LiDAR point clouds. They inherit from 2D CNNs or 3D indoor scenarios, and conduct local operators (\textit{e.g.}, SparseConv \cite{24, 25}) uniformly for all locations. This causes inferior results for the sparse distant points. As shown in Fig. 1, although decent performance is yielded for the dense close points, it is difficult for these methods to deal with the sparse distant points optimally.

We note that the root cause lies in limited receptive field. For sparse distant points, there are few surrounding neighbors. This not only results in inconclusive features, but also hinders enlarging receptive field due to information disconnection. To verify this finding, we visualize the Effective Receptive Field (ERF) \cite{40} of the given feature (shown with the yellow star) in Fig. 2 (d). The ERF cannot be expanded due to disconnection, which is caused by the extreme sparsity of the distant car.

Although window self-attention \cite{22, 30}, dilated self-attention \cite{42}, and large-kernel CNN \cite{10} have been proposed to conquer the limited receptive field, these methods do not specially deal with LiDAR point distribution, and remain to enlarge receptive field by stacking local operators as before, leaving the information disconnection issue still unsolved. As shown in Fig. 1, the method of cubic self-attention brings a limited improvement.

In this paper, we take a new direction to aggregate long-range information directly in a single operator to suit the varying-sparsity point distribution. We propose the module \textit{SphereFormer} to perceive useful information from points
50+ meters away and yield large receptive field for feature extraction. Specifically, we represent the 3D space using spherical coordinates \((r, \theta, \phi)\) with the sensor being the origin, and partition the scene into multiple non-overlapping windows. Unlike the cubic window shape, we design radial windows that are long and narrow. They are obtained by partitioning only along the \(\theta\) and \(\phi\) axis, as shown in Fig. 2 (b). It is noteworthy that we make it a plugin module to conveniently insert into existing mainstream backbones.

The proposed module does not rely on stacking local operators to expand receptive field, thus avoiding the disconnection issue, as shown in Fig. 2 (e). Also, it facilitates the sparse distant points to aggregate information from the dense-point region, which is often semantically rich. So, the performance of the distant points can be improved significantly (\(i.e., +17.1\%\) mIoU) as illustrated in Fig. 1.

Moreover, to fit the long and narrow radial windows, we propose exponential splitting to obtain fine-grained relative position encoding. The radius \(r\) of a radial window can be over 50 meters, which causes large splitting intervals. It thus results in coarse position encoding when converting relative positions into integer indices. Besides, to let points at varying locations treat local and global information differently, we propose dynamic feature selection to make further improvements.

In total, our contribution is three-fold.

- We propose SphereFormer to directly aggregate long-range information from dense-point region. It increases the receptive field smoothly and helps improve the performance of sparse distant points.
- To accommodate the radial windows, we develop exponential splitting for relative position encoding. Our dynamic feature selection further boosts performance.
- Our method achieves new state-of-the-art results on multiple benchmarks of both semantic segmentation and object detection tasks.

2. Related Work

2.1. LiDAR-based 3D Recognition

Semantic Segmentation. Segmentation \([6, 14, 15, 31, 32, 34, 49, 59–61, 83]\) is a fundamental task for vision perception. Approaches for LiDAR-based semantic segmentation can be roughly grouped into three categories, \(i.e.,\) view-based, point-based, and voxel-based methods. View-based methods either transform the LiDAR point cloud into a range view \([3, 43, 46, 68, 69]\), or use a bird-eye view (BEV) \([80]\) for a 2D network to perform feature extraction. 3D geometric information is simplified.

Point-based methods \([28, 30, 44, 45, 56, 58, 73]\) adopt the point features and positions as inputs, and design abundant operators to aggregate information from neighbors. Moreover, the voxel-based solutions \([13, 24, 25]\) divide the 3D space into regular voxels and then apply sparse convolutions. Further, methods of \([12, 17, 29, 37, 55, 71, 89]\) propose various structures for improved effectiveness. All of them focus on capturing local information. We follow this line
of research, and propose to directly aggregate long-range information.

Recently, RPVNet [70] combines the three modalities by feature fusion. Furthermore, 2DPASS [72] incorporates 2D images during training, and [48] fuses multi-modal features. Despite extra 2D information, the performance of these methods still lags behind compared to ours.

**Object Detection.** 3D object detection frameworks can be roughly categorized into single-stage [11, 26, 36, 76, 84, 85] and two-stage [19, 41, 50, 51] methods. VoxelNet [86] extracts voxel features by PointNet [44] and applies RPN [47] to obtain the proposals. SECOND [74] is efficient thanks to the accelerated sparse convolutions. Our experiments are based on CenterPoint [79], which is a widely used anchor-free framework. It is effective and efficient. Also, [65] is proposed to improve the distance objects. With similar goal, we aim to enhance the features of sparse distant points, and our proposed module can be conveniently inserted into existing frameworks.

### 2.2. Vision Transformer

Recently, Transformer [64] become popular in various 2D image understanding tasks [5, 16, 20, 21, 38, 42, 54, 62, 63, 66, 75, 71, 88]. ViT [21] tokenizes every image patch and adopts a Transformer encoder to extract features. Further, PVT [67] presents a hierarchical structure to obtain a feature pyramid for dense prediction. It also proposes Spatial Reduction Attention to save memory. Also, Swin Transformer [38] uses window-based attention and proposes the shifted window operation in the successive Transformer block. Moreover, methods of [16, 20, 75] propose different designs to incorporate long-range dependencies. There are also methods [22, 30, 42, 53, 82] that apply Transformer into 3D vision. Few of them consider the point distribution of LiDAR point cloud. In our work, we utilize the varying-sparsity property, and design radial window self-attention to capture long-range information, especially for the sparse distant points.

### 3. Our Method

In this section, we first elaborate on radial window partition in Sec. 3.1. Then, we propose the improved position encoding and dynamic feature selection in Sec. 3.2 and 3.3.

#### 3.1. Spherical Transformer

To model the long-range dependency, we adopt the window-attention [38] paradigm. However, unlike the cubic window attention [22, 30, 42], we take advantage of the varying-sparsity property of LiDAR point cloud and present the SphereFormer module, as shown in Fig. 3.

**Radial Window Partition.** Specifically, we represent LiDAR point clouds using the spherical coordinate system \((r, \theta, \phi)\) with the LiDAR sensor being the origin. We partition the 3D space along the \(\theta\) and \(\phi\) axis. We, thus, obtain a number of non-overlapping radial windows with a long and narrow 'pyramid' shape, as shown in Fig. 3. We obtain the window index for the token at \((r_i, \theta_i, \phi_i)\) as

\[
\text{win-index}_i = \left(\frac{\theta_i}{\Delta \theta}, \frac{\phi_i}{\Delta \phi}\right),
\]

where \(\Delta \theta\) and \(\Delta \phi\) denote the window size corresponding to the \(\theta\) and \(\phi\) dimension, respectively. Tokens with the same window index would be assigned to the same window. The multi-head self-attention [64] is conducted within each window independently as follows.

\[
\hat{q} = f \cdot W_q, \quad \hat{k} = f \cdot W_k, \quad \hat{v} = f \cdot W_v,
\]

where \(f \in \mathbb{R}^{n \times c}\) denotes the input features of a window, \(W_q, W_k, W_v \in \mathbb{R}^{c \times c}\) are the linear projection weights, and \(\hat{q}, \hat{k}, \hat{v} \in \mathbb{R}^{n \times c}\) are the projected features. Then, we split the projected features \(\hat{q}, \hat{k}, \hat{v}\) into \(h\) heads (i.e., \(\mathbb{R}^{n \times (h \times d)}\)), and reshape them as \(q, k, v \in \mathbb{R}^{h \times n \times d}\). For each head, we perform dot product and weighted sum as

\[
\text{attn}_k = \text{softmax}(q_k \cdot k_k^T),
\]

\[
\hat{z}_k = \text{attn}_k \cdot v_k,
\]

where \(q_k, k_k, v_k \in \mathbb{R}^{n \times d}\) denote the features of the \(k\)-th head, and \(\text{attn}_k \in \mathbb{R}^{n \times n}\) is the corresponding attention weight. Finally, we concatenate the features from all heads and apply the final linear projection with weight \(W_{proj} \in \mathbb{R}^{c \times c}\) to yield the output \(z \in \mathbb{R}^{n \times c}\) as

\[
\hat{z} = \text{concat}(\{\hat{z}_0, \hat{z}_1, \ldots, \hat{z}_{h-1}\}).
\]

\[
z = \hat{z} \cdot W_{proj}.
\]
SparseConvNet [24, 25], MinkowskiNet [13], local window self-attention [22, 30, 42]. In this paper, we find that inserting it into the end of each stage works well, and the network structure is given in the supplementary material. The resulting model can be applied to various downstream tasks, such as semantic segmentation and object detection, with strong self-attention [22, 30]. In this paper, we find that inserting it into the end of each stage works well, and the network structure is given in the supplementary material. The resulting model can be applied to various downstream tasks, such as semantic segmentation and object detection, with strong self-attention [22, 30].

3.2. Position Encoding

For the 3D point cloud network, the input features have already incorporated the absolute xyz position. Therefore, there is no need to apply absolute position encoding. Also, we notice that Stratified Transformer [30] develops the contextual relative position encoding. It splits a relative position into several discrete parts uniformly, which converts the continuous relative positions into integers to index the positional embedding tables.

This method works well with local cubic windows. But in our case, the radial window is narrow and long, and its radius $r$ can take even more than 50 meters, which could cause large intervals during discretization and thus coarse-grained positional encoding. As shown in Fig. 4 (a), because of the large interval, $key_1$ and $key_2$ correspond to the same index. But there is still a considerable distance between them.

**Exponential Splitting.** Specifically, since the $r$ dimension covers long distances, we propose exponential splitting for the $r$ dimension as shown in Fig. 4 (b). The splitting interval grows exponentially when the index increases. In this way, the intervals near the query are much smaller, and the $key_1$ and $key_2$ can be assigned to different position encodings. Meanwhile, we remain to adopt the uniform splitting for the $\theta$ and $\phi$ dimensions. In notation, we have a query token $q_i$ and a key token $k_j$. Their relative position $(r_{ij}, \theta_{ij}, \phi_{ij})$ is converted into integer index $(idx^r_{ij}, idx^\theta_{ij}, idx^\phi_{ij})$ as

$$
idx^r_{ij} = \begin{cases} 
- \max(0, \lceil \log_2(e^{-r_{ij} \theta}) \rceil) - 1 & r_{ij} < 0 \\
\max(0, \lceil \log_2(e^{r_{ij} \phi}) \rceil) & r_{ij} > 0 
\end{cases},
$$

$$
idx^\theta_{ij} = \lfloor \frac{\theta_{ij}}{\text{interval}_\theta} \rfloor, \quad idx^\phi_{ij} = \lfloor \frac{\phi_{ij}}{\text{interval}_\phi} \rfloor,
$$

where $a$ is a hyper-parameter to control the starting splitting interval, and $L$ is the length of the positional embedding tables. Note that we also add the indices with $\frac{L}{2}$ to make sure they are non-negative.

The above indices $(idx^r_{ij}, idx^\theta_{ij}, idx^\phi_{ij})$ are then used to index their positional embedding tables $t_r, t_\theta, t_\phi \in \mathbb{R}^{L \times (h \times d)}$ to find the corresponding position encoding $p^r_{ij}, p^\theta_{ij}, p^\phi_{ij} \in \mathbb{R}^{h \times d}$, respectively. Then, we sum them up to yield the resultant positional encoding $p \in \mathbb{R}^{h \times d}$, which then performs dot product with the features of $q_i$ and $k_j$, respectively. The original Eq. (3) is updated to

$$
p = p^r_{ij} + p^\theta_{ij} + p^\phi_{ij},
$$

$$
pos_bias_{k,i,j} = q_{k,i} \cdot p^r_{ij} + k_{k,j} \cdot p^\theta_{ij} + \text{pos_bias}_{k,i,j},
$$

$$
\text{attn}_{k} = \text{softmax}(q_{k,i} \cdot k_{k,j}^T + \text{pos_bias}_{k,i,j}),
$$

where $\text{pos_bias} \in \mathbb{R}^{h \times n \times n}$ is the positional bias to the attention weight, $q_{k,i} \in \mathbb{R}^d$ means the the $k$-th head of the $i$-th query feature, and $p_k \in \mathbb{R}^d$ is the $k$-th head of the position encoding $p$. The exponential splitting strategy provides smaller splitting intervals for near token pairs and larger intervals for distant ones. This operation enables a fine-grained position representation between near token pairs, and still maintains the same number of intervals in the meanwhile. Even though the splitting intervals become larger for distant token pairs, this solution actually works well since distant token pairs require less fine-grained relative position.
3.3. Dynamic Feature Selection

Point clouds scanned by LiDAR have the varying-sparcity property — close points are dense and distant points are much sparser. This property makes points at different locations perceive different amounts of local information. For example, as shown in Fig. 5, a point of the car (circled in green) near the LiDAR is with rich local geometric information from its dense neighbors, which is already enough for the model to make a correct prediction — incurring more global contexts might be contrarily detrimental. However, a point of bicycle (circled in red) far away from the LiDAR lacks shape information due to the extreme sparsity and even occlusion. Then we should supply long-range contexts as a supplement. This example shows treating all the query points equally is not optimal. We thus propose to dynamically select local or global features to address this issue.

As shown in Fig. 6, for each token, we incorporate not only the radial contextual information, but also local neighbor communication. Specifically, input features are projected into query, key and value features as Eq. (2). Then, the first half of the heads are used for radial window self-attention, and the remaining ones are used for cubic window self-attention. After that, these two features are concatenated and then linearly projected to the final output z for feature fusion. It enables different points to dynamically select local or global features. Formally, the Equations (3-5) are updated to

$$\text{attn}_k^{radial} = \text{softmax}(q_k^{radial} \cdot k_k^{radialT}),$$

$$z_k^{radial} = \text{attn}_k^{radial} \cdot v_k^{radial},$$

$$\text{attn}_k^{cubic} = \text{softmax}(q_k^{cubic} \cdot k_k^{cubicT}),$$

$$z_k^{cubic} = \text{attn}_k^{cubic} \cdot v_k^{cubic},$$

$$\hat{z}_k = \text{concat}((z_0^{radial}, z_1^{radial}, \ldots, z_{h/2-1}^{radial}, z_h^{cubic}, \ldots, z_{h-1}^{cubic})).$$
The learning rate and weight decay are set to 0.006 and 0.01, respectively. Batch size is set to 16 on nuScenes, and 8 on both SemanticKITTI and Waymo Open Dataset. The window size is set to 0.9.$\theta$. Batch size is set to 16 on nuScenes, and 8 on both SemanticKITTI and Waymo Open Dataset. During data preprocessing, we confine the input scene to the range $[-51.2 \text{m}, -51.2 \text{m}, -4 \text{m}]$ to $[51.2 \text{m}, 51.2 \text{m}, 2 \text{m}]$ on SemanticKITTI and $[-75.2 \text{m}, -75.2 \text{m}, -2 \text{m}]$ to $[75.2 \text{m}, 75.2 \text{m}, 4 \text{m}]$ on Waymo. Also, we set the voxel size to 0.1m on both nuScenes and Waymo, and 0.05m on SemanticKITTI.

For object detection, we adopt the OpenPCDet [57] codebase and follow the default CenterPoint [79] to set the training hyper-parameters. We set the window size to $[120 \text{m}, 1.5^\circ, 1.5^\circ]$.

### 4.2. Semantic Segmentation Results

The results on SemanticKITTI test set are shown in Table 1. Our method yields 74.8% mIoU, a new state-of-the-art result. Compared to the methods based on range images [43, 68] and Bird-Eye-View (BEV) [80], ours gives a result with over 20% mIoU performance gain. Moreover, thanks to the capability of directly aggregating long-range information, our method significantly outperforms the models based on sparse convolution [12, 55, 70, 71, 89]. It is also notable that our method outperforms 2DPASS [72] that uses extra 2D images in training by 1.9% mIoU.

In Tables 2 and 3, we also show the semantic segmentation results on nuScenes test set. Methods published before the submission deadline (11/11/2022) are listed.
### 4.4. Ablation Study

To testify the effectiveness of each component, we conduct an extensive ablation study and list the result in Table 5. The Experiment I (Exp. I for short) is our baseline model of SparseConv. Unless otherwise specified, we train the models on nuScenes train set and make evaluations on nuScenes val set for the ablation study. To comprehensively reveal the effect, we also report the performance at different distances, i.e., close ($≤20m$), medium ($>20m$ and $≤50m$), far ($>50m$) distances.

#### Window Shape

By comparing Experiments I and II in Table 5, we can conclude that the radial window shape is beneficial. Further, the improvement stems mainly from better handling the medium and far points, where we yield 5.67% and 13.39% mIoU performance gain, respectively. This result exactly verifies the benefit of aggregating long-range information with the radial window shape.

Moreover, we also compare the radial window shape with the cubic one proposed in [22, 30, 42]. As shown in Table 6, the radial window shape considerably outperforms the cubic one.

Besides, we investigate the effect of window size as shown in Table 7. Setting it too small may make it hard to capture meaningful information, while setting it too large to note that our method is purely based on LiDAR data, and it works even better than approaches of [23, 72, 90] that use additional 2D information.

Moreover, we demonstrate the semantic segmentation results on Waymo Open Dataset val set in Table 4. Our model outperforms the baseline model with a substantial gap of 3.3% mIoU. Also, it is worth noting that our method achieves a 9.3% mIoU performance gain for the far points, *i.e.*, the sparse distant points.

#### 4.3. Object Detection Results

Our method also achieves strong performance in object detection. As shown in Table 8, our method outperforms other published methods on nuScenes test set, and ranks 3rd on the LiDAR-only benchmark. It shows that directly aggregating long-range information is also beneficial for object detection. It also manifests the capability of our method to generalize to instance-level tasks.

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
<th>barrier</th>
<th>bicycle</th>
<th>bus</th>
<th>car</th>
<th>construction</th>
<th>motorcycle</th>
<th>pedestrian</th>
<th>traffic-cone</th>
<th>truck</th>
<th>driveable</th>
<th>other-flat</th>
<th>sidewalk</th>
<th>terrain</th>
<th>manmade</th>
<th>vegetation</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeNet53++ [43]</td>
<td>65.5</td>
<td>66.0</td>
<td>21.3</td>
<td>77.2</td>
<td>80.9</td>
<td>30.2</td>
<td>66.8</td>
<td>69.6</td>
<td>52.1</td>
<td>54.2</td>
<td>72.3</td>
<td>94.1</td>
<td>66.6</td>
<td>63.5</td>
<td>70.1</td>
<td>83.1</td>
</tr>
<tr>
<td>PolarNet [80]</td>
<td>71.0</td>
<td>74.7</td>
<td>28.2</td>
<td>85.3</td>
<td>90.9</td>
<td>35.1</td>
<td>77.5</td>
<td>71.3</td>
<td>58.8</td>
<td>57.4</td>
<td>76.1</td>
<td>96.5</td>
<td>71.1</td>
<td>74.7</td>
<td>74.0</td>
<td>87.3</td>
</tr>
<tr>
<td>Salsanext [18]</td>
<td>72.2</td>
<td>74.8</td>
<td>34.1</td>
<td>85.9</td>
<td>88.4</td>
<td>42.2</td>
<td>72.4</td>
<td>72.2</td>
<td>63.1</td>
<td>61.3</td>
<td>76.5</td>
<td>96.0</td>
<td>70.8</td>
<td>71.2</td>
<td>71.5</td>
<td>86.7</td>
</tr>
<tr>
<td>AMVNet [35]</td>
<td>76.1</td>
<td>79.8</td>
<td>32.4</td>
<td>82.2</td>
<td>86.4</td>
<td>62.5</td>
<td>81.9</td>
<td>75.3</td>
<td>72.3</td>
<td>83.5</td>
<td>65.1</td>
<td>97.4</td>
<td>67.0</td>
<td>78.8</td>
<td>74.6</td>
<td>90.8</td>
</tr>
<tr>
<td>Cylinder3D [89]</td>
<td>76.1</td>
<td>76.4</td>
<td>40.3</td>
<td>91.2</td>
<td>93.8</td>
<td>51.3</td>
<td>78.0</td>
<td>78.9</td>
<td>64.9</td>
<td>62.1</td>
<td>84.4</td>
<td>96.8</td>
<td>71.6</td>
<td>76.4</td>
<td>75.4</td>
<td>90.5</td>
</tr>
<tr>
<td>PVKD [27]</td>
<td>76.0</td>
<td>76.2</td>
<td>40.0</td>
<td>90.2</td>
<td>94.0</td>
<td>50.9</td>
<td>77.4</td>
<td>78.8</td>
<td>64.7</td>
<td>62.0</td>
<td>84.1</td>
<td>96.6</td>
<td>71.4</td>
<td>76.4</td>
<td>76.3</td>
<td>90.3</td>
</tr>
<tr>
<td>RPVNet [70]</td>
<td>77.6</td>
<td>78.2</td>
<td>43.4</td>
<td>92.7</td>
<td>93.2</td>
<td>49.0</td>
<td>80.5</td>
<td>66.0</td>
<td>66.9</td>
<td>84.0</td>
<td>96.9</td>
<td>73.5</td>
<td>75.9</td>
<td>76.0</td>
<td>90.6</td>
<td>88.9</td>
</tr>
<tr>
<td>Ours</td>
<td>78.4</td>
<td>77.7</td>
<td>43.8</td>
<td>94.3</td>
<td>93.1</td>
<td>52.4</td>
<td>86.9</td>
<td>81.2</td>
<td>65.4</td>
<td>73.4</td>
<td>85.3</td>
<td>97.0</td>
<td>73.4</td>
<td>75.4</td>
<td>75.0</td>
<td>91.0</td>
</tr>
<tr>
<td>Ours†</td>
<td>79.5</td>
<td>78.7</td>
<td>46.7</td>
<td>95.2</td>
<td>93.7</td>
<td>54.0</td>
<td>88.9</td>
<td>81.1</td>
<td>68.0</td>
<td>74.2</td>
<td>86.2</td>
<td>97.2</td>
<td>74.3</td>
<td>76.3</td>
<td>75.8</td>
<td>91.4</td>
</tr>
</tbody>
</table>

Table 3. Semantic segmentation results on nuScenes val set. † denotes using rotation and translation testing-time augmentations.

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
<th>close</th>
<th>med</th>
<th>far</th>
<th>overall</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparseConv [25]</td>
<td>66.6</td>
<td>67.8</td>
<td>64.1</td>
<td>52.6</td>
<td>94.4 85.1</td>
<td>37.8 2.2 89.1 73.4 40.4 74.8 57.3 66.6 75.2 95.5 91.3 67.0 68.1 92.3 41.7 30.1 79.0 75.6</td>
</tr>
<tr>
<td>Ours</td>
<td>69.9</td>
<td>70.3</td>
<td>68.6</td>
<td>61.9</td>
<td>94.5 86.7</td>
<td>40.2 0.9 69.7 90.2 73.9 41.8 77.2 65.4 71.9 83.7 95.9 91.7 68.4 69.8 93.3 53.9 47.9 80.8 77.2</td>
</tr>
</tbody>
</table>

Table 4. Semantic segmentation results on Waymo Open Dataset val set.

<table>
<thead>
<tr>
<th>ID</th>
<th>RadialWin</th>
<th>ExpSplit</th>
<th>Dynamic</th>
<th>close</th>
<th>medium</th>
<th>far</th>
<th>overall</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>78.79</td>
<td>51.54</td>
<td>13.28</td>
<td>75.21</td>
<td>0.00</td>
</tr>
<tr>
<td>II</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>78.95</td>
<td>57.21</td>
<td>26.67</td>
<td>76.31</td>
<td>+1.10</td>
</tr>
<tr>
<td>III</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>79.92</td>
<td>61.09</td>
<td>31.10</td>
<td>77.60</td>
<td>+2.39</td>
</tr>
<tr>
<td>IV</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>79.51</td>
<td>58.94</td>
<td>28.95</td>
<td>77.05</td>
<td>+1.84</td>
</tr>
<tr>
<td>V</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>80.80</td>
<td>60.78</td>
<td>30.38</td>
<td>78.41</td>
<td>+3.20</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>close</th>
<th>medium</th>
<th>far</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubic</td>
<td>79.21</td>
<td>54.31</td>
<td>19.31</td>
<td>76.19</td>
</tr>
<tr>
<td>Radial</td>
<td>80.80</td>
<td>60.78</td>
<td>30.38</td>
<td>78.41</td>
</tr>
</tbody>
</table>

Table 6. Comparison between radial and cubic window shapes.

<table>
<thead>
<tr>
<th>window size</th>
<th>mIoU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0°</td>
<td>77.8</td>
</tr>
<tr>
<td>1.5°</td>
<td>77.5</td>
</tr>
<tr>
<td>2.0°</td>
<td>78.4</td>
</tr>
<tr>
<td>2.5°</td>
<td>77.6</td>
</tr>
</tbody>
</table>

Table 7. Effect of window size for the $θ$ and $ϕ$ dimensions.

To test the effectiveness of each component, we conduct an extensive ablation study and list the result in Table 5. The Experiment I (Exp. I for short) is our baseline model of SparseConv. Unless otherwise specified, we train the models on nuScenes train set and make evaluations on nuScenes val set for the ablation study. To comprehensively reveal the effect, we also report the performance at different distances, *i.e.*, close (*≤20m*), medium (*>20m* and *≤50m*), far (*>50m*) distances.
Table 8. Object detection results on nuScenes test set. Methods published before the submission deadline (11/11/2022) are listed.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PointPillars [33]</td>
<td>45.3</td>
<td>30.5</td>
<td>68.4</td>
<td>23.0</td>
<td>28.2</td>
<td>23.4</td>
<td>4.1</td>
<td>59.7</td>
<td>27.4</td>
<td>1.1</td>
<td>30.8</td>
<td>38.9</td>
</tr>
<tr>
<td>3DSSD [77]</td>
<td>56.4</td>
<td>42.6</td>
<td>81.2</td>
<td>47.2</td>
<td>61.4</td>
<td>30.5</td>
<td>12.6</td>
<td>70.2</td>
<td>36.0</td>
<td>8.6</td>
<td>31.1</td>
<td>47.9</td>
</tr>
<tr>
<td>CBSG [87]</td>
<td>63.3</td>
<td>52.8</td>
<td>81.1</td>
<td>48.5</td>
<td>54.9</td>
<td>42.9</td>
<td>10.5</td>
<td>80.1</td>
<td>51.5</td>
<td>22.3</td>
<td>70.9</td>
<td>65.7</td>
</tr>
<tr>
<td>CenterPoint [79]</td>
<td>65.5</td>
<td>58.0</td>
<td>84.6</td>
<td>51.0</td>
<td>60.2</td>
<td>53.2</td>
<td>17.5</td>
<td>83.4</td>
<td>53.7</td>
<td>28.7</td>
<td>76.7</td>
<td>70.9</td>
</tr>
<tr>
<td>HotSpotNet [8]</td>
<td>66.0</td>
<td>59.3</td>
<td>83.1</td>
<td>50.9</td>
<td>56.4</td>
<td>53.3</td>
<td>23.0</td>
<td>81.3</td>
<td>63.5</td>
<td>36.6</td>
<td>73.0</td>
<td>71.6</td>
</tr>
<tr>
<td>CVCGNET [7]</td>
<td>70.2</td>
<td>65.5</td>
<td>86.2</td>
<td>56.7</td>
<td>66.3</td>
<td>58.8</td>
<td>28.2</td>
<td>86.1</td>
<td>68.3</td>
<td>44.2</td>
<td>82.0</td>
<td>78.2</td>
</tr>
<tr>
<td>TransFusion [2]</td>
<td>70.0</td>
<td>63.8</td>
<td>86.7</td>
<td>56.3</td>
<td>67.7</td>
<td>59.5</td>
<td>23.8</td>
<td>87.5</td>
<td>64.5</td>
<td>36.3</td>
<td>81.4</td>
<td>74.1</td>
</tr>
<tr>
<td>FocalConv [9]</td>
<td>72.8</td>
<td>68.5</td>
<td>84.9</td>
<td>55.1</td>
<td>66.4</td>
<td>59.3</td>
<td>29.9</td>
<td>86.0</td>
<td>71.4</td>
<td>47.1</td>
<td>79.7</td>
<td>75.2</td>
</tr>
<tr>
<td>Ours‡</td>
<td>72.8</td>
<td>68.5</td>
<td>85.3</td>
<td>57.9</td>
<td>67.0</td>
<td>59.9</td>
<td>33.7</td>
<td>88.6</td>
<td>76.3</td>
<td>56.4</td>
<td>82.2</td>
<td>78.2</td>
</tr>
</tbody>
</table>

‡ Flipping and rotation testing-time augmentations.

Figure 7. Visual comparison between vanilla SparseConv and ours (best viewed in color and by zoom-in). The brown box is the zoom-in of the cyan box. The last two columns are the difference maps with the ground truth. More examples are given in the supplementary material.

4.5. Visual Comparison

As shown in Fig. 7, we visually compare the baseline model (i.e., SparseConv) and ours. It visually indicates that with our proposed module, more sparse distant objects are recognized, which are highlighted with cyan boxes. More examples are given in the supplementary material.

5. Conclusion

We have studied and dealt with varying-sparsity LiDAR point distribution. We proposed SphereFormer to enable the sparse distant points to directly aggregate information from the close ones. We designed radial window self-attention, which enlarges the receptive field for distant points to intervene with close dense ones. Also, we presented exponential splitting to yield more detailed position encoding. Dynamically selecting local or global features is also helpful. Our method demonstrates powerful performance, ranking 1st on both nuScenes and SemanticKITTI semantic segmentation benchmarks and achieving the 3rd on nuScenes object detection benchmark. It shows a new way to further enhance 3D visual understanding. Our limitations are discussed in the supplementary material.
References


[34] Yiming Li, Tao Kong, Ruixiang Chu, Yifeng Li, Peng Wang, and Lei Li. Simultaneous semantic and collision learning for 6-dof grasp pose estimation. In IROS. IEEE, 2021.


[70] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. Rpvnnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. In ICCV, 2021. 1, 3, 6, 7

[71] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguang Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In AAAI, 2021. 1, 2, 6

[72] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In ECCV, 2022. 1, 3, 6, 7


[74] Yan Yan, Yuxing Mao, and Bo Li. SECOND: sparsely embedded convolutional detection. Sensors, 18(10):3337, 2018. 3


[76] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In CVPR, pages 11037–11045, 2020. 3

[77] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In CVPR, 2020. 8


[90] Zhuangwei Zhuang, Rong Li, Kui Jia, Qicheng Wang, Yuanqing Li, and Mingkui Tan. Perception-aware multi-sensor fusion for 3d lidar semantic segmentation. In ICCV, 2021. 6, 7